الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE MATHÉMATIQUES

SÉRIE 01

OBJECTIF PÉDAGOGIQUE :

À la fin de cette série, le stagiaire doit être capable d'étudier les systèmes de numérotation et l'algèbre de Boole et de résoudre les problèmes de calcul matériel.

PLAN DE LA LEÇON :

I- LES SYSTÈMES DE NUMÉROTATION :

1- Base d'un système de numérotation :

Nous avons pris l'habitude de représenter les nombres en utilisant dix symboles différents: 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9

Ce système est appelé le système **décimal** (déci signifie dix).

Il existe cependant d'autres formes de numération qui fonctionnent en utilisant un nombre de symboles distincts.

Exemple :

Système binaire (bi: deux),
Le système octal (oct: huit),
Le système hexadécimal (hexa: seize).

En fait, on peut utiliser n'importe quel nombre de symboles différents (pas nécessairement des chiffres).

Dans un système de numération : le nombre de symboles distincts est appelé **la base** du système de numération

Pour compter des objets et les représenter par des nombres, on utilise des "systèmes de numération", en général "pondérés".

Système de numération →
- Base du système
- Symboles (digits)
- Poids du digit selon son rang

2- Les différents systèmes de numérotations :

1.2- Système Décimal :

La base dix est très ancienne. Elle découle d'un choix naturel, dicté par le nombre des doigts des deux mains :

Pour le système de numérotation décimal on a :

Base : 10

Symboles :0,1,2,3,4,5,6,7,8,9

Poids: puissance de 10

3 4558 2

Poids fort

Poids faible

345 , 567

Partie décimal

Partie fractionnelle

Développement en polynôme d'un nombre dans le système décimal

Soit le nombre 2011, ce nombre peut être écrit sous la forme suivante :

$2011= 2*10^{3} + 0*10^{2} + 1*10^{1} + 1*10^{0}$
$2011= 2*1000+ 0*100 +1*10+ 1*1$
$2011= 2000+ 000+ 10+ 1$

Cette forma s'appelle la forme polynomiale
Un nombre réel peut être écrit aussi sous la forme polynomiale

$1978,265 = 1*10^{3} + 9*10^{2} + 7*10^{1} + 8*10^{0} + 2*10^{-1} + 6*10^{-2} + 5*10^{-3}$

Comptage en décimal

- Sur une seule position : 0, 1, 2, 3, 4, 5,….9= 10^{1}-1
- Sur deux positions : 00, 01,02, …..,99=10^{2}-1
- Sur trois positions 000,001,……,999=10^{3}-1
- Sur n positions : minimum 0,.., maximum 10^{n-1}, nombre de Combinaisons 10^{n}

2.2- Système Binaire :

C’est le système utilisé par les ordinateurs pour faire des calculs et communiquer.

Pour le système de numération binaire on a :

- Base : 2
- Symboles : 0,1
- Poids: puissance de 2

Dans le système binaire, pour exprimer n’importe quelle valeur on utilise uniquement 2 symboles : {0, 1}

$(1\ \ 0\ \ 010\ 1)_2$

Un bit

La base

$(1\ 0010\ 1)_2$

Le Bit du poids fort

Le Bit du poids faible

Remarque :

Un nombre binaire de **4 bit** est appelé **quartet**.

Exemple : 1010

Un nombre binaire de **8 bit** est appelé **octet**.

Exemple : 10011110

Un nombre dans la base 2 peut être écrit aussi sous la forme polynomiale.

$(1110)_2 = 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = (14)_{10}$

$(1110,101)2 = 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = (14,625)_{10}$

Comptage en binaire :

Sur un seul bit : 0, 1

Sur deux bits

Binaire	**Décimal**
00	0
01	1
10	2
11	3

4

Combinaisons= 2^2

Sur trois bits

Binaire	Décimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

8

Combinaisons= 2^3

3.2- Système Octal :

Ce système permet d'abréger l'écriture des nombres binaires.

Pour le système de numération octal on a :

- Base : 8
- Symboles : 0, 1, 2, 3, 4, 5, 6, 7
- Poids: puissance de 8

8 symboles sont utilisés dans ce système: {0, 1, 2, 3, 4, 5, 6, 7}

Exemple 1 :

$(127)_8 = 1 * 8^2 + 2 * 8^1 + 7 * 8^0 = (87)_{10}$

$(127{,}65)_8 = 1 * 8^2 + 2 * 8^1 + 7 * 8^0 + 6 * 8^{-1} + 5 * 8^{-2} = (87{,}828125)_{10}$

Exemple 2 :

Le nombre $(1289)_8$ n'existe pas dans la base 8 puisque les symboles 8 et 9 n'appartiennent pas à la base.

4.2- Système Hexadécimal :

Ce système permet d'abréger l'écriture des nombres binaires.

Pour le système de numération hexadécimal on a :

- Base : 16
- Symboles :0,1,2,3,4,5,6,7,8,9,A ,B ,C,D,E,F
- Poids: puissance de 16

16 symboles sont utilisés dans ce système:{ 0, 1, 2, 3, 4, 5, 6, 7, 8,9, A, B, C, D, E, F}

$(17)_{16} = 1 * 16^1 + 7 * 16^0 = (23)_{10}$
$(AB)_{16} = A * 16^1 + B * 16^0 = 10 * 16^1 + 11 * 16^0 = (171)_{10}$

Hexadécimal	Décimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

II- CONVERSION D'UNE BASE X À LA BASE 10 :

Cette conversion est assez simple puisque il suffit de faire le développement en polynôme de ce nombre dans la base X, et de faire la somme par la suite.

$(1101)_2= 1*2^3 +1*2^2 +0*2^1+ 1*2^0 =(13)_{10}$

$(1A7)_{16}=1*16^2+A*16^1+7*16^0=1*16+10*16+7*16=$

$256+160+7=(423)_{10}$

$(1101,101)_2= 1*2^3+ 1*2^2+0*2^1+1*2^0 +1*2^{-1}+ 0*2^{-2}+ 1*2^{-3}=$

$(13,625)_{10}$

$(43,2)_5= 4*5^1+ 3*5^0+ 2*5^{-1} =20+ 3+ 0,4 =(23,4)_{10}$

RESUME :

Dans une base X, on utilise X symboles distincts pour représenter les nombres.

La valeur de chaque symbole doit être strictement inférieure à la base X.
Chaque nombre dans une base X peut être écrit sous sa forme polynomiale.

III- CONVERSION ENTRE BASES :

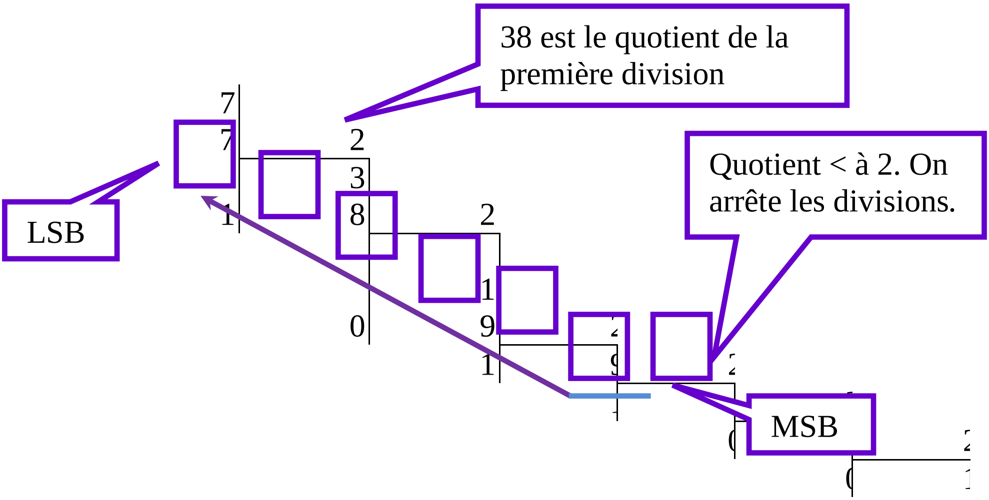
1- <u>Conversion Décimal – Binaire</u> :

Pour passer du décimal vers une le binaire :

On divise le nombre à convertir par la base d'arrivée (2).
On répète les divisions tant que le quotient est supérieur ou égal à la base (2).

Le résultat est donné en lisant le dernier quotient et les restes de la dernière vers la première division.

<u>Exemple</u> : convertir 77 en binaire.



Lecture du résultat

$(77)_{10} = (1001101)_2$

Conversion de la base 10 à la base 2 : cas d'un nombre réel

Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle.

La partie entière est transformée en effectuant des divisions successives.

La partie fractionnelle est transformée en effectuant des multiplications successives par 2.

Exemple1 : $(77{,}625)_{10} = (\ ?)_2$

Partie Entière=$(77)_{10} = (1001101)_2$

Partie Fractionnelle = $(0{,}625)_{10} = (?)_2$

$$
\begin{array}{lll}
0{,}625 * 2 & = & \boxed{1},25 \\
0{,}25 * 2 & = & \boxed{0},5 \\
0{,}5 * 2 & = & \boxed{1},0
\end{array}
$$

$(0{,}625)_{10} = (0{,}101)_2$

Donc $(77{,}625)_{10} = (77{,}101)_2$

Exemple 2 : $(0{,}6)_{10} = (\ ?)_2$

$$
\begin{array}{lll}
0{,}6 * 2 & = & 1{,}2 \\
0{,}2 * 2 & = & 0{,}4 \\
0{,}4 * 2 & = & 0{,}8 \\
0{,}8 * 2 & = & 1{,}6
\end{array}
\longrightarrow \quad (0{,}6)_{10} = (0{,}1001)_2
$$

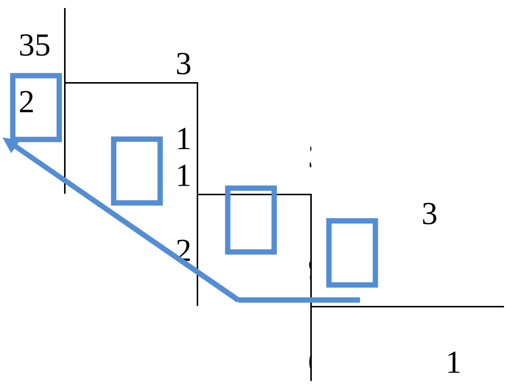
REMARQUE : Le nombre de bits après la virgule va déterminer la précision.

2- Conversion du décimal à une base X :

La conversion se fait en prenant les restes des divisions successives sur la base X dans le sens inverse.

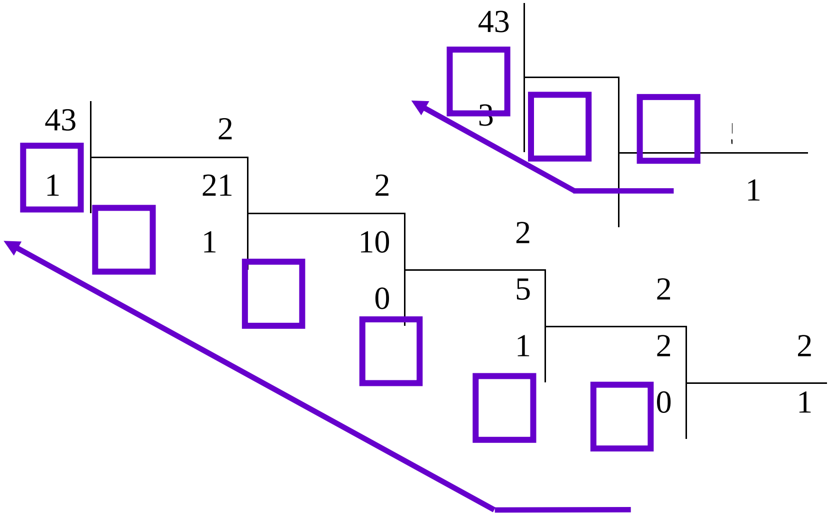
Exemple1 : $(35)_{10}=(?)_3$.

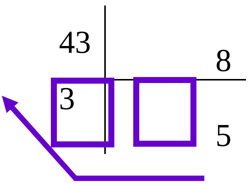
$(35)_{10}=(1022)_3$.

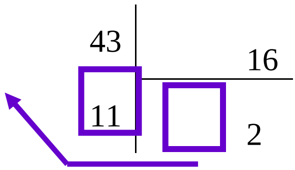


QUESTION : Effectuer les transformations suivantes :

$(43)_{10}=(?)_2=(?)_5=(?)_8=(?)_{16}$.





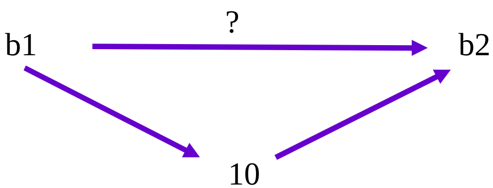


$(43)_{10}=(101011)_2=(133)_5=(53)_8= (2B)_{16}$.

Conversion d'une base b1 à une base b2 :

Il n'existe pas de méthode pour passer d'une base **b1** à une autre base **b2** directement.

L'idée est de convertir le nombre de la base **b1** à la base 10, en suite convertir le résultat de la base 10 à la base **b2**.



Exemple1 : $(34)_5=(?)_7$.

$$(34)_5=3*5^1+4*5^0=15+4=(19)_{10}=(25)_7$$

$$\begin{array}{r|ll} 19 & & 7 \\ \hline \boxed{5} & \boxed{2} & 2 \end{array}$$

3- Conversion Binaire – Octal –Hexadécimal et vice versa :

Conversion Binaire – Octal

En octal chaque, symbole de la base s’écrit **sur 3 bits en binaire**.

L’idée de base est de replacer chaque symbole dans la base octal par sa valeur en binaire sur 3bits (faire des éclatements sur 3 bits).

Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Exemples :

$(345)_8=(011\ 100\ 101)_2$
$(65,76)_8=(110\ 101,\ 111\ 110)_2$
$(35,34)_8=(011\ 101,\ 011\ 100)_2$

Remarque :

Le remplacement se fait de droite à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle.

Conversion Octal – Binaire

L’idée de base est de faire des **regroupements** de **3 bits** à partir du poids faible.

Par la suite **remplacer** chaque regroupement par la valeur octal correspondante.

Exemple :

$(11001010010110)_2 = (011\ 001\ 010\ 010\ 110)_2 = (31226)_8$

$(110010100,10101)_2 = (110\ 010\ 100,\ 101\ 010)_2 = (624,51)_8$

Remarque :

Le regroupement se fait de droite à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle.

Conversion Binaire – Hexadécimal

En hexadécimal chaque, symbole de la base s'écrit **sur 4 bits en binaire**.

L'idée de base est de replacer chaque symbole dans la base octal par sa valeur en binaire sur 4 bits (faire des éclatements sur 4 bits).

Octal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Exemples :

$(345B)_{16} = (0011\ 0100\ 0101\ 1011)_2$

$(AB3,\ 4F6)_{16} = (1010\ 1011\ 0011,\ 0100\ 1111\ 0110)_2$

Conversion Hexadécimal – Binaire

L'idée de base est de faire des **regroupements** de **4 bits** à partir du poids faible.

Par la suite **remplacer** chaque regroupement par la valeur hexadécimal correspondante.

Exemple :

$(11001010100110)_2 = (0011\ 0010\ 1010\ 0110)_2 = (32A6)_{16}$
$(110010100{,}10101)_2 = (0001\ 1001\ 0100{,}1010\ 1000)_2 = (194{,}\ A8)_{16}$

IV- OPÉRATEURS ARITHMÉTIQUES EN BINAIRE :

1- Addition :

L'addition binaire s'effectue de la même manière qu'une addition décimale, mais en se situant dans l'arithmétique des nombres binaires. Pour ce faire, on commencera par poser la table d'addition binaire :

Exemple 1 :

A	B	A+ B	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$(1010101010010011)_2 + (100010001101101)_2 = (\ ?)_2$

1 1 1 1 1

Retenues

A +   1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1

B   0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 1

1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0

Donc

$(1010101010010011)_2 + (100010001101101)_2 = (1110111100000000)_2$

Exemple2 :

(10101110100100011)$_2$+(100110001101101)$_2$=(?)$_2$

Retenues

A		1 0 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 1
B	+	0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 1 1 0 1
Somme		1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0

1010111010010011)$_2$+(100110001101101)$_2$=(1111101100000000)$_2$

Exemple2 : (10101110)$_2$+(11111000)$_2$=(?)$_2$

Retenues			1	1	1	1				
A			1	0	1	0	1	1	1	0
B	+		1	1	1	1	1	0	0	0
Somme		1	1	0	1	0	0	1	1	0

Donc (10101110)$_2$+(11111000)$_2$=(110100110)$_2$

2- Soustraction :

La soustraction binaire s'effectue de la même manière qu'une soustraction décimale, on commencera par poser la table de soustraction binaire :

A	B	A- B	E
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Exemple : $(110101101)_2 - (101100110)_2 = (\ ?)_2$

A		1	1	0	1	0	1	1	0	1
B	-	1	0	1	1	0	0	1	1	0
Emprunt			1				1	1		
Soustraction		0	0	1	0	0	0	1	1	1

Donc $(110101101)_2 - (101100110)_2 = (1000111)_2$

3-Multiplication :

La multiplication binaire s'effectue de la même manière qu'une multiplication décimale, on commencera par poser la table de multiplication binaire :

A	B	A x B
0	0	0
0	1	0
1	0	0
1	1	1

Exemple : $(1101)_2 \times (101)_2 = (\ ?)_2$

A					1	1	0	1
B	x					1	0	1
	=				1	1	0	1
	+			0	0	0	0	.
	+		1	1	0	1	.	.
	=	1	0	0	0	0	0	1

Donc $(1101)_2 \times (101)_2 = (1000001)_2$

V- DIVISION :

La division binaire est le reflet exact de la division décimale. On utilise une nouvelle fois les mêmes méthodes, et les mêmes propriétés s'appliquent.

Exemple : $(10100101)_2$ / $(101)_2$=(?$)_2$

1	0	1	0	0	1	0	1	1	0	1	1
	1	0	1	1				1	1	1	1
	1	0	0	1	1						
		1	0	1	1						
	0	1	0	0	0	0					
			1	0	1	1					
		0	0	1	0	1	1				
				1	0	1	1				
				0	0	0	0				

Donc $(10100101)_2$ / $(1011)_2$=$(1111)_2$

VI- LA COMPLÉMENTATION :

Le **complément** permet de coder des nombres négatifs. En utilisant n bits, on peut alors représenter les nombres de – $(2^{n-1} -1)$ à + $2^{n-1} -1$: le premier bit pour le signe (0 : positif, 1 : négatif) et n-1 bits pour le nombre.

Exemple : Sur 8 bits on peut représenter -127 à +127
Sur 4 bits on peut représenter -7 à +7

Le complément existe sous deux formes : complément à un, et complément à deux.

1- Complément à un :

1	1	1	0	1	0	0	0
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-128	64	32	0	8	0	0	0
=	-24	+1=	-23				

Le complément à 1 d'un nombre binaire est trouvé en changeant tous les 1 à 0 et tous les 0 à 1

1 0 1 1 0 0 1 0

0 1 0 0 1 1 0 1

Exemple :

$+5=(0101)_2$ sur 4 bits → $-5=(1010)_2$ sur 4 bits
$+8=(001010)_2$ sur 6 bits → $-8=(110101)_2$ sur 6 bits
$+15=(00001111)_2$ sur 8 bits → $-15=(11110000)_2$ sur 8 bits

ÉVALUATION :

- Pour Les nombres positifs sont évalués en évaluant la magnitude comme représentation binaire du nombre.
- Pour les nombres négatifs : On affecte une valeur négative au poids du signe de bit, Et On somme les poids dont la valeur du bit est à 1 Et on ajoute un 1

Exemple : $(11101000)_2=(\ ?)_{10}$ sur 8 bits

2- Complément à deux :

Cette méthode est la seule utilisable mathématiquement, Elle permet une utilisation des nombres signés avec une représentation unique du zéro et la possibilité d'effectuer des calculs.

Le complément à deux est trouvé en ajoutant un 1 au bit le moins significatif (LSF) du complément à 1

– Complément à 2 = (Complément à 1) + 1

10110010 → 01001101+1
→ 01001110

Exemple :

$+5=(0101)_2$ sur 4 bits → $-5=(1011)_2$ sur 4 bits
$+8=(001010)_2$ sur 6 bits → $-8=(110110)_2$ sur 6 bits
$+15=(00001111)_2$ sur 8 bits → $-15=(11110001)_2$ sur 8 bits

ÉVALUATION :

- Les nombres positifs et négatifs sont évalués en faisant la somme des poids correspondant à des bits valant 1
- Attention : On affecte une valeur négative au poids du signe de bit pour un nombre négatif

Exemple 1 : $(01010110)_2=(\ ?)_{10}$ sur 8 bits

0	1	0	1	0	1	1	0
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	64	0	16	0	4	2	0
=	+86						

Exemple 2 : $(10101010)_2=(\ ?)_{10}$ sur 8 bits

1	0	1	0	1	0	1	0
-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-128	0	32	0	8	0	2	0
=	-86						

3- Soustraction par complément à deux et addition :

- Addition :

Quatre cas sont possibles :

Les deux nombres sont positifs : Dans ce cas-là on effectue une addition binaire classique.

Exemple : $(00100011)_2+(00100111)_2=(\ ?)_2$ sur 8 bits

Retenues			1			1	1	1	
A		0	0	1	0	0	0	1	1
B	+	0	0	1	0	0	1	1	1
Somme		0	1	0	0	1	0	1	0

$(00100011)_2+(00100111)_2=(01001010)_2$

Le nombre positif est plus 'grand' que le nombre négatif : On effectue une addition de binaire classique, On 'oublie' la dernière retenue (à gauche), La somme est positive.

Exemple : $(01100011)_2+(11010100)_2=(\ ?)_2$ sur 8 bits

Retenues									
A		0	1	1	0	0	0	1	1
B	+	1	1	0	1	0	1	0	0
Somme		0	0	1	1	0	1	1	1

$(01100011)_2+(11010100)_2=(00110111)_2$

b- Le nombre négatif est plus 'grand' que le nombre positif : On effectue une addition binaire classique, La somme est négative et représentée directement dans le système complément à 2.

c-Exemple : $(00100011)_2+(11010100)_2=(\ ?)_2$ sur 8 bits

Retenues									
A		0	0	1	0	0	0	1	1
B	+	1	1	0	1	0	1	0	0
Somme		1	1	1	1	0	1	1	1

$(00100011)_2+(11010100)_2=(11110111)_2$

d- Les deux nombres sont négatifs : On effectue une addition binaire classique, On oublie la dernière retenue la plus à gauche, Et le résultat est négatif et déjà représenté dans le système complément à 2

Exemple : $(10100011)_2+(11110100)_2=(\ ?)_2$ sur 8 bits

Retenues									
A		1	0	1	0	0	0	1	1
B	+	1	1	1	1	0	1	0	0
Somme		1	0	0	1	0	1	1	1

$(10100011)_2+(11110100)_2=(10010111)_2$
Dépassement de capacité (Overflow)

On parle **d'Overflow** quand il y a dépassement de la capacité pour représenter le résultat d'une somme

- Quand il y a **Overflow** la somme n'est pas de même signe que les opérandes
- Cela peut se produire uniquement si les deux opérandes sont de mêmes signes

Exemple : $(01111101)_2+(00111010)_2=(?)_2$ sur 8 bits

Retenues		1	1	1	1				
A		0	1	1	1	1	1	0	1
B	+	0	0	1	1	1	0	1	0
Somme		1	0	1	1	0	1	1	1

$(01111101)_2+(00111010)_2=(10110111)_2$ sur 8 bits

- Soustraction :

La soustraction est considérée comme un cas particulier
de l'addition :

A - B = A + (-B)
-A - B = (-A) + (-B)

On prend donc le système complément à deux pour représenter (-B),
Et on effectue une addition.

Exemple : $(01100101)_2 - (00101011)_2=(?)_2$ sur 8 bits
$(01100101)_2 - (00101011)_2= (01100101)_2 + (11010101)_2$

Retenues						1		1	
A		0	1	1	0	0	1	0	1
B	+	1	1	0	1	0	1	0	1
Somme		0	0	1	1	1	0	1	0

$(01100101)_2 - (00101011)_2=(00111010)_2$

EXERCICES D'APPLICATION :

EXERCICE N°01 :

Effectuer les transformations suivantes à la base 10 ?

$(123)_6=(?)_{10}$
$(45,76)_8=(?)_{10}$
$(1100,11)_2=(?)_{10}$
$(1ABC)_{16}=(?)_{10}$

EXERCICE N°02 :

Effectuer les transformations suivantes :

$(23,65)_{10}=(?)_2$
$(18,190)_{10}=(?)_2$

EXERCICE N°03 :

Effectuer les transformations suivantes

$(43)_6=(?)_5=(?)_8$
$(2A)_{16}=(?)_9$

EXERCICE N°04 :

Effectuer les opérations suivantes

$(1111101)_2+(1110101)_2=(?)_2$
$(10001011)_2+(1110101)_2=(?)_2$
$(1111101)_2-(1110101)_2=(?)_2$
$(10001011)_2-(1110101)_2=(?)_2$
$(11001)_2x(10001)_2=(?)_2$
$(100111)_2x(10101)_2=(?)_2$
$(1110101)_2 / (1101)_2=(?)_2$
$(1110001111101)_2 / (1101)_2=(?)_2$

EXERCICE N°05 :

Effectuer les transformations en compléments à deux des nombres décimaux suivantes (sur 8 puis 10 bits).

13, -15 , 122 , 255, -100, 128

EXERCICE N°06 :

Effectuer les opérations suivantes

$(11001)_2+(10001)_2=(?)_2$	sur 8 bits
$(100111)_2 - (10101)_2=(?)_2$	sur 6 bits
$(01101101)_2+(11110101)_2=(?)_2$	sur 8 bits
$(11001011)_2+(1000101)_2=(?)_2$	sur 10 bits
$(1111101)_2-(1110101)_2=(\ ?)_2$	sur 7 bits
$(10001011)_2-(1110101)_2=(?)_2$	sur 8 bits

CORRIGÉ DES EXERCICES :

EXERCICE N°01 :

$(123)_6=1*6^2+2*6^1+3*6^0=36+12+3=(51)_{10}$

$(45,76)_8=4*8^1+5*8^0+2*8^{-1}+2*8^{-2}=32+5+0,25+0,015625=(37,265625)_{10}$

$(1100,11)_2=1*2^3+1*2^2+0*2^1+0*2^0+1*2^{-1}+1*2^{-2}=$
$8+4+0+0+0,5+0,25=(12,75)_{10}$

$(1ABC)_{16}$
$=1*16^3+A*16^2+B*16^1+C*16^0=1*4096+10*256+11*16+12*1=$
$4096+2560+176+12=(6844)_{10}$

EXERCICE N°02 :

$(23,65)_{10}=(10111,101001)_2$

$(18,190)_{10}=(10010,00110000101)_2$

EXERCICE N°03 :

$(43)_6=(27)_{10}=(102)_5=(33)_8$

$(2A)_{16}=(42)_{10}=(46)_9$

EXERCICE N°04 :

$(1111101)_2+(1110101)_2=(11110010)_2$

$(10001011)_2+(1110101)_2=(100000000)_2$

$(1111101)_2-(1110101)_2=(1000)_2$

$(10001011)_2-(1110101)_2=(10110)_2$

$(11001)_2x(10001)_2=(110101001)_2$

$(100111)_2x(10101)_2=(1100110011)_2$

$(1110101)_2 / (1101)_2=(1001)_2$

$(1110001111101)_2 / (1101)_2=(1000110001)_2$

EXERCICE N°5 :

	En 8 bits	En 10 bits
13	00001101	0000001101
-15	11110001	1111110001
122	01111010	0001111010
255	impossible	0011111111
-100	10011100	1110011100
128	impossible	0010000000

EXERCICE N°6 :

Effectuer les opérations suivantes

$(11001)_2+(10001)_2=(00101010)_2$ sur 8 bits

$(100100)_2 - (10101)_2=(001111)_2$ **Overflow** sur 6 bits

$(01101101)_2+(11110101)_2=(01100010)_2$ sur 8 bits

$(11001011)_2+(1000101)_2=(0100010000)_2$ sur 10 bits

$(1111101)_2-(1110101)_2=(0001000)_2$ sur 7 bits

$(10001011)_2-(1110101)_2=(10000000)_2$ sur 8 bits